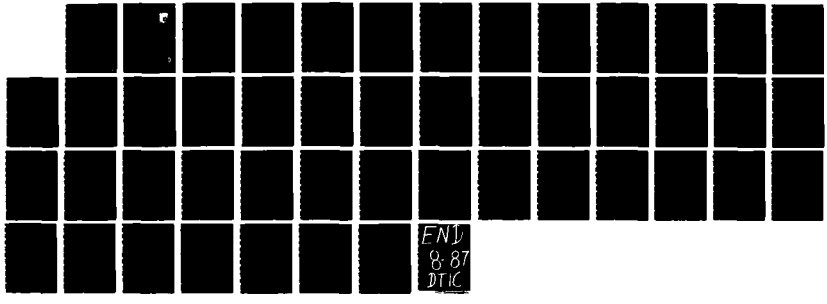
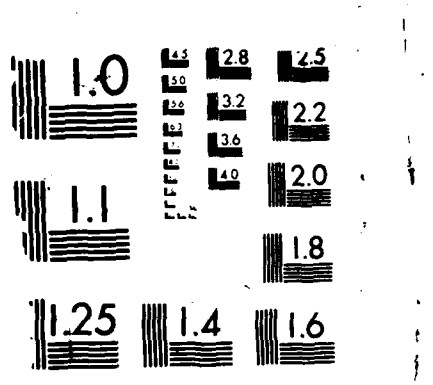
AD-A182 651 INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 8 1/1
USER INTERFACE SUBS (U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONSU
UNCLASSIFIED C MORENC ET AL 01 NOV 85 DS-620144502 F/G 12/5 NL

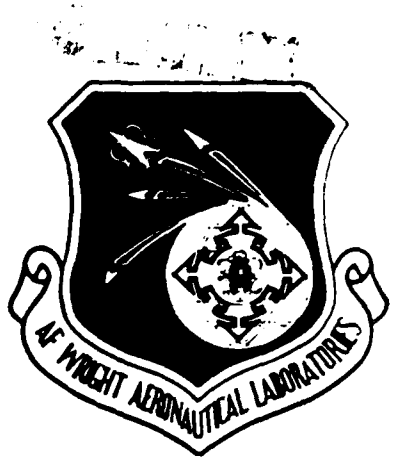




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AD-A182 651

AFWAL-TR-86-4006
Volume VIII
Part 24



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 24 - Rapid Application Generator Development Specification

General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345

Final Report for Period 22 September 1980 - 31 July 1985

November 1985

Approved for public release; distribution is unlimited.

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533

DTIC
ELECTE
JUL 16 1987
D

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS) At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

DAVID L. JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986
DATE

FOR THE COMMANDER:

GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86
DATE

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MLTC, W-PAFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

A182 651

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-4006 Vol VIII, Part 24
6a NAME OF PERFORMING ORGANIZATION General Electric Company Production Resources Consulting 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION AFWAL/MLTC
6c ADDRESS (City, State and ZIP Code) 1 River Road Schenectady, NY 12345	7b ADDRESS (City, State and ZIP Code) WPAFB, OH 45433-6533
8a NAME OF FUNDING/SPONSORING ORGANIZATION Materials Laboratory Air Force Systems Command, USAF 8b OFFICE SYMBOL (If applicable) AFWAL/MLTC	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-80-C-5155
8c ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433	10 SOURCE OF FUNDING NOS PROGRAM ELEMENT NO. 78011F; PROJECT NO. 7500; TASK NO. 62; WORK UNIT NO. 01
11 TITLE (Include Security Classification) (See Reverse)	
12 PERSONAL AUTHOR(S) Butterworth, Manning and Morenc, Carol and Robie, Penny	
13a TYPE OF REPORT Final Technical Report	13b TIME COVERED 22 Sept 1980 - 31 July 1985
14 DATE OF REPORT (Yr., Mo., Day) 1985 November	15 PAGE COUNT 49
16 SUPPLEMENTARY NOTATION ICAM Project Priority 6201	The computer software contained herein are theoretical and/or references that in no way reflect Air Force-owned or -developed computer software.

17 COSATI CODES		
FIELD	GROUP	SUB GR
1308	0905	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This specification establishes the development, test, qualification, and performance requirements of a computer program identified as the Rapid Application Generator (RAP). The RAP is used to translate textual definitions of interactive database applications into programs that access data bases via the Common Data Model (CDM)

The application definitions are compiled by the RAP through calls to FLAN. The Application Definition Language in which the database applications are expressed includes the Forms Definition Language and other statement types.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b TELEPHONE NUMBER (Include Area Code) 513-255-6976 22c OFFICE SYMBOL AFWAL/MLTC

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

11. Title

Integrated Information Support System (IISS)
Vol VIII - User Interface Subsystem
Part 24 - Rapid Application Generator Development
Specification

A S D 86 0039
9 Jan 1986

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

QUALITY INSPECTED 4

PREFACE

This development specification covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Reviewer.
D. Appleton Company (DACOM)	Responsible for IDEF support, state-of-the-art literature search.
General Dynamics/ Ft. Worth	Responsible for factory view function and information models.

Subcontractors	Role
Illinois Institute of Technology	Responsible for factory view function research (IITRI) and information models of small and medium-size business.
North American Rockwell	Reviewer.
Northrop Corporation	Responsible for factory view function and information models.
Pritsker and Associates	Responsible for IDEF2 support.
SofTech	Responsible for IDEF0 support.

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Responsible for consultation on applications of the technology and on IBM computer technology.
Computer Technology Associates (CTA)	Assisted in the areas of communications systems, system design and integration methodology, and design of the Network Transaction Manager.
Control Data Corporation (CDC)	Responsible for the Common Data Model (CDM) implementation and part of the CDM design (shared with DACOM).
D. Appleton Company (DACOM)	Responsible for the overall CDM Subsystem design integration and test plan, as well as part of the design of the CDM (shared with CDC). DACOM also developed the Integration Methodology and did the schema mappings for the Application Subsystems.

Subcontractors	Role
Digital Equipment Corporation (DEC)	Consulting and support of the performance testing and on DEC software and computer systems operation.
McDonnell Douglas Automation Company (McAuto)	Responsible for the support and enhancements to the Network Transaction Manager Subsystem during 1984/1985 period.
On-Line Software International (OSI)	Responsible for programming the Communications Subsystem on the IBM and for consulting on the IBM.
Rath and Strong Systems Products (RSSP) (In 1985 became McCormack & Dodge)	Responsible for assistance in the implementation and use of the MRP II package (PIOS) that they supplied.
SofTech, Inc.	Responsible for the design and implementation of the Network Transaction Manager (NTM) in 1981/1984 period.
Software Performance Engineering (SPE)	Responsible for directing the work on performance evaluation and analysis.
Structural Dynamics Research Corporation (SDRC)	Responsible for the User Interface and Virtual Terminal Interface Subsystems.

Other prime contractors under other projects who have contributed to Test Bed Technology, their contributing activities and responsible projects are as follows:

Contractors	ICAM Project	Contributing Activities
Boeing Military Aircraft Company (BMAC)	1701, 2201, 2202	Enhancements for IBM node use. Technology Transfer to Integrated Sheet Metal Center (ISMC).

Contractors	ICAM Project	Contributing Activities
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP).
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology.
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements.
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI).
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

		Page
SECTION 1.0	SCOPE	1-1
1.1	Identification	1-1
1.2	Functional Summary	1-1
SECTION 2.0	DOCUMENTS	2-1
2.1	Reference Documents	2-1
2.2	Terms and Abbreviations	2-3
SECTION 3.0	REQUIREMENTS	3-1
3.1	Computer Program Definition	3-1
3.1.1	System Capacities	3-1
3.1.2	Interface Requirements	3-1
3.1.2.1	Interface Block Diagram	3-1
3.1.2.2	Detailed Interface Definition	3-3
3.2	Detailed Function Requirements	3-3
3.2.1	Application Definition	3-4
3.2.2	User Interaction	3-4
3.2.2.1	Switching Between Functions	3-4
3.2.2.2	Exception Conditions	3-4
3.2.2.3	Styles of Interaction	3-4
3.2.2.3.1	Function Key Selection	3-4
3.2.2.3.2	Menu Picking	3-5
3.2.2.3.3	Cursor Location	3-5
3.2.2.3.4	Icon Selection	3-5
3.2.3	Database Operations	3-6
3.2.3.1	Deletion	3-6
3.2.3.2	Insertion	3-6
3.2.3.3	Modification	3-7
3.2.3.4	Selection	3-7
3.2.4	Database Integrity	3-7
3.2.5	Data Display	3-8
3.2.5.1	Current Action Display	3-8
3.2.5.2	Graphical Display	3-8
3.2.5.2.1	Graph Definition	3-9
3.2.5.2.2	Additive Versus Absolute Display	3-9
3.2.5.2.3	Axis Information	3-10
3.2.5.2.4	Background Color	3-10
3.2.5.2.5	Curve Information	3-11
3.2.5.2.6	Grids	3-11
3.2.5.2.7	Legends	3-11
3.2.5.2.8	Margins	3-11
3.2.5.2.9	Pie Charts	3-12
3.2.5.2.10	Text	3-12

3.2.5.3 Stretchy Lines 3-12
3.2.6 Security And Access Control 3-13
3.2.7 Conditional Actions 3-13
3.2.7.1 Form Processor Actions 3-13
3.2.7.1.1 PRESENT Forms 3-13
3.2.7.1.2 SET Option 3-14
3.2.7.1.3 HELP Text and Forms 3-14
3.2.7.1.4 EXIT 3-14
3.2.7.2 Database Transactions 3-14
3.2.7.2.1 Delete 3-14
3.2.7.2.2 Insert 3-14
3.2.7.2.3 Modify 3-14
3.2.7.2.4 Select 3-14
3.2.8 Data Structures 3-15
3.2.9 Generated Program Structure 3-15
3.3 Performance Requirements 3-17
3.3.1 Programming Methods 3-17
3.3.2 Program Organization 3-17
3.4 Data Base Requirements 3-17
3.4.1 Sources and Types of Input 3-17
3.4.2 Destinations and Types of Outputs 3-17

SECTION 4.0 QUALITY ASSURANCE PROVISIONS 4-1
4.1 Introduction and Definitions 4-1
4.2 Computer Programming Test and Evaluation 4-1

SECTION 5.0 PREPARATION FOR DELIVERY 5-1

APPENDIX A Application Generator Language Syntax A-1

FIGURES

3-1 Application Generator Interfaces 3-2
3-2 Rapid Application Generator Data Structures 3-16

SECTION 1

SCOPE

1.1 Identification

This specification establishes the development, test, qualification, and performance requirements of a computer program identified as the Rapid Application Generator (RAP). The RAP is one configuration item of the Integrated Information Support System (IISS) User Interface (UI).

1.2 Functional Summary

This Computer Program Configuration Item (CPCI) is used to facilitate development of screen driven interactive programs accessing databases through the Common Data Model (CDM).

The major functions of the Rapid Application Generator are:

1. Provide a screen driven interface to database application programs.

2. Provide a screen based means of displaying the contents of a database.

3. Provide a context within which a single application program can switch between modes of database access: update, query, deletion, and insertion.

4. Allow the application developer to apply human engineering to the means by which the user dialogues with the database program.

SECTION 2

DOCUMENTS

2.1 Reference Documents

[1] Structural Dynamics Research Corporation, IISS Terminal Operator Guide, OM 620144000 , 1 November 1985.

[2] Structural Dynamics Research Corporation, IISS Form Processor User Manual, UM 620144200B, 1 November 1985.

[3] General Electric Company, System Design Specification, 7 February 1983.

[4] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS 620144401B, 1 November 1985.

[5] Structural Dynamics Research Corporation, Forms Driven Form Editor Development Specification, DS 620144402B, 1 November 1985.

[6] Structural Dynamics Research Corporation, Report Writer Development Specification, DS 620144501 , 1 November 1985.

[7] Structural Dynamics Research Corporation, Rapid Application Generator Development Specification, DS 620144502 , 1 November 1985.

[8] Structural Dynamics Research Corporation, Text Editor Development Specification, DS 620144600B, 1 November 1985.

[9] Structural Dynamics Research Corporation, Application Interface Development Specification, DS 620144700 , 1 November 1985.

[10] Structural Dynamics Research Corporation, User Interface Services Development Specification, DS 620144100B, 1 November 1985.

[11] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620144200B, 1

November 1985.

[12] Structural Dynamics Research Corporation, Virtual Terminal Interface Development Specification, DS 620144300B, 1 November 1985.

[13] Systran, ICAM Documentation Standards, 15 September 1983.

[14] Systran, User's Manual for the ICAM Integrated Support System (IISS) Neutral Data Manipulation Language (NDML), February, 1983.

[15] Systran, Implementation of Enhancements of NDML SELECT COMMAND, 25 July 1984, revised 9 September 1984.

[16] Systran, Discussion of Function Implementation NDML SELECT COMMAND, 25 July 1984, revised 4 September 1984.

2.2 Terms and Abbreviations

Application Definition Language: an extension of the Forms Definition Language that includes retrieval of database information and conditional actions. It is used to define interactive application programs.

Application Generator: (AG), subset of the IISS User Interface that consists of software modules that generate IISS application code and associated form definitions based on a language input. The part of the AG that generates report programs is called the Report Writer. The part of the AG that generates interactive applications is called the Rapid Application Generator.

Application Interface: (AI), subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process: (AP), a cohesive unit of software that can be initiated as a unit to perform some function or functions.

Attribute: field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Common Data Model: (CDM), IISS subsystem that describes common data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Communication Services: allows on host interprocess communication and inter-host communication between the various Test Bed subsystems.

Communication Subsystem: (COMM), IISS subsystem that provides communication services to the Test Bed and subsystems.

Computer Program Configuration Item: (CPCI), an aggregation of computer programs or any of their discrete portions, which

satisfies an end-use function.

Conceptual Schema: (CS), the standard definition used for all data in the CDM. It is based on IDEF1 information modelling.

Cursor Position: the position of the cursor after any command is issued.

Device Drivers: (DD), software modules written to handle I/O for a specific kind of terminal. The modules map terminal specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: is similar to the open list, except that it contains only those forms that have been added to the screen and are currently displayed on the screen.

External Schema: (ES), an application's view of the CDM's conceptual schema.

Field: two-dimensional space on a terminal screen.

Field Pointer: indicates the ITEM which contains the current cursor position.

Form: structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows.

Form Definition: (FD), forms definition language after compilation. It is read at runtime by the Form Processor.

Forms Definition Language: (FDL), the language in which electronic forms are defined.

Forms Driven Form Editor: (FDFE), subset of the FE which consists of a forms driven application used to create Form Definition files interactively.

Form Editor: (FE), subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor and the Forms Language Compiler.

Form Hierarchy: a graphic representation of the way in which forms, items and windows are related to their parent form.

Forms Language Compiler: (FLAN), subset of the FE that consists of a batch process that accepts a series of forms definition language statements and produces form definition files as output.

Form Processor: (FP), subset of the IISS User Interface that consists of a set of callable execution time routines available to an application program for form processing.

Form Processor Text Editor: (FPTE), subset of the Form Processor that consists of software modules that provide text editing capabilities to all users of applications that use the Form Processor.

IISS Function Screen: the first screen that is displayed after logon. It allows the user to specify the function he wants to access and the device type and device name on which he is working.

Integrated Information Support System: (IISS), a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item: non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output.

Logical Device: a conceptual device which to an application is indistinguishable from a physical device and is then mapped to part or all of a physical device.

Message: descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: a line on the terminal screen that is used to display messages.

Network Transaction Manager: (NTM), IISS subsystem that performs the coordination, communication and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Neutral Data Manipulation Language: (NDML), the command language by which the CDM is accessed for the purpose of extracting, deleting, adding, or modifying data.

Open List: a list of all the forms that have been and are currently open for an application process.

Operating System: (OS), software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: instance of forms in windows that are created whenever a form is added to a window.

Paging and Scrolling: a method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Physical Device: a hardware terminal.

Presentation Schema: (PS), may be equivalent to a form. It is the view presented to the user of the application.

Qualified Name: the name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

Rapid Application Generator: (RAP), part of the Application Generator that generates source code for interactive programs based on a language input.

Subform: a form that is used within another form.

Text Editor: (TE), subset of the IISS User Interface that consists of a file editor that is based on the text editing functions built into the Form Processor.

User Data: data which is either input by the user or output by the application programs to items.

User Interface: (UI), IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System: (UIDS), collection of

IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor and the Application Generator.

User Interface Management System: (UIMS), the runtime UI. It consists of the Form Processor, Virtual Terminal, Application Interface, the User Interface Services and the Text Editor.

User Interface Monitor: (UIM), part of the Form Processor that handles messaging between the NTM and the UI. It also provides authorization checks and initiates applications.

User Interface Services: (UIS), subset of the IISS User Interface that consists of a package of routines that aid users in controlling their environment. It includes message management, change password, and application definition services.

User Interface/Virtual Terminal Interface: (UI/VTI), another name for the User Interface.

Virtual Terminal: (VT), subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.

Virtual Terminal Interface: (VTI), the callable interface to the VT.

Window: dynamic area of a terminal screen on which predefined forms may be placed at run time.

Window Manager: a facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor.

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

The Rapid Application Generator is used to translate textual definitions of interactive database applications into programs that access data bases via the CDM.

The application definitions are compiled by the Rapid Application Generator through calls to FLAN. The Application Definition Language in which the database applications are expressed includes the Forms Definition Language and other statement types.

3.1.1 System Capacities

The RAP is written partially in C and in Cobol. For every application generated by the RAP the following are produced:

1) C code for controlling screen input/output

2) Cobol code for NDML calls

3) Form Definition Objects

4) Header files for External Schemas and Presention Schemas

3.1.2 Interface Requirements

The COBOL program output by the RAP is constrained to be compatible with statement forms expected by the CDM precompiler.

The syntax of the Application Definition Language accepted as input to FLAN is modelled after the Forms Definition Language and the Neutral Data Manipulation Language.

3.1.2.1 Interface Block Diagram

The interface block diagram for the Application Rapid Generator is shown in Figure 3-1.

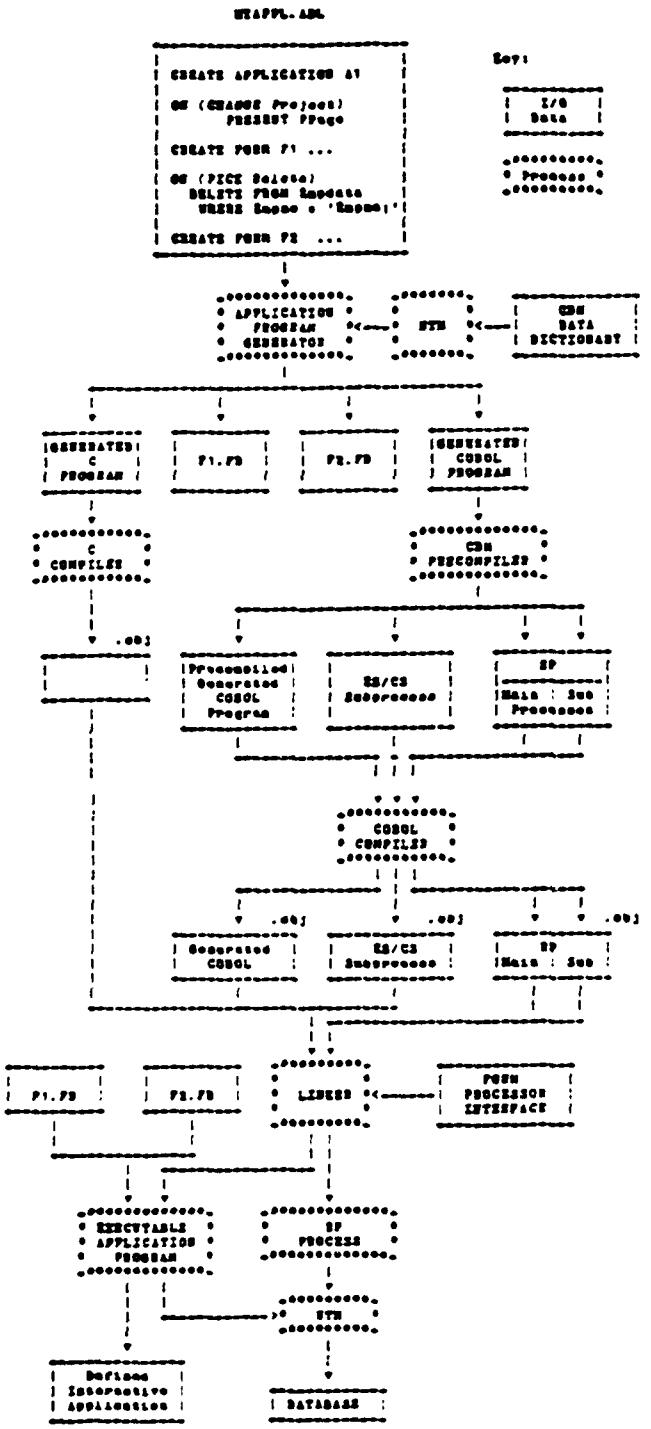


Figure 3-1 Rapid Application Generator Interfaces

3.1.2.2 Detailed Interface Definition

The syntax of the Rapid Application Definition Language is documented in Appendix A. This language is intended to provide access to all interactive Application Generator functionality.

The collection of ADL entries that define a database application is an application definition. An application definition is written to an ADL source file with any text editor one might use to prepare a program source file.

The ADL source file is processed by the Application Generator to produce separate binary form files for each form definition and to generate C and COBOL code which together with the Form Processor routines and data dictionary information contained in the Common Data Model provide the final interactive application.

Since the COBOL program contains NDML statements to access the database via the CDM, it must be preprocessed by the CDM precompiler. This step will not only produce a COBOL compilable version of the generated routine, it will also produce a conceptual to external schema transform subroutine and a query program. After compilation and linking, the generated C and COBOL programs are executed to produce the on-line database application.

3.2 Detailed Function Requirements

The RAP exploits the versatility of the Forms Definition Language and extends it in ways needed to define the user/program dialogue. The RAP produces the application by accessing the CDM and outputting the data using form processor routines.

When a processing error occurs, an error message will be output to the user's terminal. The error message will include as much information as possible about the nature of the error and steps necessary to correct it. If at all possible, processing should continue after an error is recognized so that additional errors may also be detected.

When the user has initiated a transaction which would extensively modify the database, the application should warn the user and request confirmation before committing the operation.

3.2.1 Application Definition

The application is defined by using the application definition language which subsumes the forms definition language.

3.2.2 User Interaction

Applications implemented with the Rapid Application Generator are controlled in their execution by the direction of the user at run time. The manner in which this can occur is described below.

3.2.2.1 Switching Between Functions

The Rapid Application Generator shall offer the application developer the ability to switch between database operations, e.g. delete or insert, while remaining within a single application. The function switching shall be user input driven.

3.2.2.2 Exceptional Conditions

The user's interaction, via the terminal, with the application program comprises a series of events. These events can be detected by the application program and predefined actions can be triggered in consequence. It is these user-initiated events which determine the course of execution. These events include cursor positioning, function key selection, change of value of an item field, or the occurrence of a certain value in an item field.

3.2.2.3 Styles of Interaction

3.2.2.3.1 Function Key Selection

The function keys definable to the VTI may be assigned arbitrary named functions used subsequently to define a detectable event. Customarily certain of the program function keys are assigned a priori meanings of mode, help, quit, and so on. Others could be defined to mean delete, insert, next, etc. In this way the user can signal an event which could be a forms processor operation or a database transaction; or, a single key stroke can result in several actions. The outcome of the event is prescribed by the application definition.

3.2.2.3.2 Menu Picking

Menu picking can take one of several forms and all are available to the application developer. In fact, several can be combined in a single application although this could be confusing to a user and diminish the effectiveness of the application. The cursor location can be determined at the time a function key is depressed. A choice from a predefined collection of alternatives can be detected by establishing an event for the occurrence of a certain value in an item field. Or, a menu pick can be indicated simply by the occurrence of any non-blank value in a field.

3.2.2.3.3 Cursor Location

This mode of interaction is a general case of one of the menu pick methods. The difference is that the event which the user is signalling is not customarily considered a menu selection. The cursor location is monitored and combined with the depression of a function key. An example of this style of interaction is the selection of one or more records to delete from a database. The selection could be indicated by positioning the cursor within the record form.

3.2.2.3.4 Icon Selection

The Application Generator shall provide the ability by selection of an icon to choose operations to perform and objects to manipulate. The association of the icon with the operation to be performed is specified in the application definition as are the location and size of the icon.

The icon itself, that is its pictorial representation, may be defined by some item field which in general is not displayed or it may be defined explicitly by a bit or character string pattern given as part of the icon field definition. The size associated with the icon is the size of a containing rectangle surrounding the icon and is given in units of the standard character height and width on the display device. The icon may be detected by location of the cursor, or optionally some other locator mechanism within the containing rectangle.

Detection of an icon can be monitored by an ON Condition described elsewhere and can result in one or more actions being performed.

There are many ways in which an iconic figure could be defined. Only the very simplest is specified here. The method described here is that of indicating on an element by element basis what appears within each portion of the icon field. What are individually addressable as elements within the field will depend upon the display device. The field is always addressable on a character position by character position basis. In this mode the icon may be defined by a string formed from available characters (including graphic characters). Alternatively, the icon may be defined by a bit map of the pixels within the field. Zero indicates the pixel has the attributes of the background. The character or pixel maps are given in row-major order, (i.e., row by row).

3.2.3 Database Operations

In one of the abovementioned ways a developer can allow the application user to perform one or several of the following database operations. All database transactions are ultimately performed via NDML; thus, the functionality offered by the RAP is the same as that provided through embedded NDML.

3.2.3.1 Deletion

The Rapid Application Generator will allow an application developer to define database delete operations equivalent to those provided by embedded NDML. These include the deletion of records for which specified fields satisfy designated conditions. These conditions can either be implicit as when a displayed record is marked for deletion and the condition is implied to be the equality of the displayed column fields with the displayed values or explicit as when a specified item field is used as the condition and is assigned a character string value which expresses the condition. If the selection criterion is explicit or the delete operation will cause removal of records other than those displayed, then the application may request confirmation of the action before its execution. The verification may simply be a general affirmative response, or it may be a confirmation for each record including perhaps the display of the record's contents.

3.2.3.2 Insertion

The insert operation supported by the RAP is that supported by NDML. Thus, the result of an insertion is the addition of new rows at the end of the original table.

3.2.3.3 Modification

Modifications may be made to one or more fields of one or more rows of a table. The rows to be modified are determined by a condition which may be implicit or explicit. The condition is implicit when it is the equality of values of displayed fields and the values of corresponding record fields in the database. It is explicit when the condition is taken from an item field that contains a Boolean expression. In general, modifications are made by actually altering the contents of the displayed item fields. If the selection criteria are implicit, the previous values of the item fields are used to identity the database records to be modified. If the selection criterion is explicit, the new record field value is taken from a user input value in a displayed item field.

When modification would be made to more than a single record or to records not displayed, the application may request confirmation from the user before committing the transaction to the database. The confirmation can be a general affirmative response or individual confirmation for each record to be modified.

3.2.3.4 Selection

The database may also be queried to display record contents on the user's terminal. The application definition determines what subset of all the records' fields are actually displayed. The selection can either be performed by giving an actual predicate expression or by giving values to selected item fields which then become "key" fields of the database. The key fields defined in the application need not be synonymous with the actual database key fields; they simply determine what records are extracted for display. It is possible that more than one database record could project onto the same set of displayed field values.

3.2.4 Database Integrity

The Rapid Application Generator does not insure database integrity.

3.2.5 Data Display

3.2.5.1 Current Action Display

In many cases it will be helpful to the user to have one or more database records displayed at the same time as the database operation is performed. For example, display and modification of form fields corresponding to record fields provides a full-screen database editor. The RAP shall provide this capability plus scrolling as well. The RAP also performs ES to PS and PS to ES conversion. The display attributes are determined by PICTURE clauses, if present, and by the user's input.

3.2.5.2 Graphical Display

The Rapid Application Generator will provide the capability to graphically display data in two-dimensional format using common business graphics. The RAP will provide this through a field, the graph field, which may contain a single graph of predefined type and structure. The available types are bar chart, pie chart, and x-y plot. In the case of bar charts and x-y plots more than one data set may be plotted in a single graph yielding multiple curves or parallel or stacked bars, for example.

Size and location values are in terms of the default terminal character sizes and positions. No scaling is done automatically; it is the user's responsibility to ensure that textual information, for example, is of the proper size to fit in the available space. Portions of a graph which do not fit in a displayed space are clipped, not wrapped. The RAP will attempt to adjust graphs appropriately for the aspect ratio of the device on which they are displayed so that, for example, pie charts are circles regardless of the display device.

Optional attributes will have reasonable default values so that a graph can be included in a display with a minimum of effort at specification. Attributes unsupported on a given device, for example color on a monochrome device, will default to appropriate values.

On devices supporting them, capabilities shall include the choice of line style and width (e.g. solid, dashed, dotted, thin, and thick), and the choice of color of the curve, the background, and the area under the curve. Other choices shall always be available including the specification of the size and

font style of text in axis labels, tick mark labels, and legend entries, the lower and upper limits of axes, the scale (linear or logarithmic) of the axes, the presence or absence of a grid, the sizes of the margins between the borders of the graph field and the graph, some shading pattern below the curves, symbols to appear at the data points, and for pie charts the amount of explosion of individual segments as well as the appearance and location (inside or outside) of the actual numerical values and/or the percentages represented by each segment. For graphs consisting of more than one curve or more than one set of bars there shall also be the option of displaying the data additively or absolutely as measured on the vertical axis.

Graphs shall be for output display only. In particular, although the cursor position may be detected within the graph field, no smaller portion of the graph field may be picked, nor may any data points be picked or altered interactively.

The data sources for the graphs denoted by ordinate and/or abscissa in the language syntax are numeric repeating item fields. The item fields may or may not be displayed as form fields. If they are input fields, then data in them can be altered and the next time the graph using those fields is displayed it will reflect the new values. Graph fields cannot be presented in windows because the information defining them appears in part in the field definition of the CREATE FORM statement and in part in the CREATE GRAPH statement.

The following paragraphs explain the language features describing graphs in somewhat more detail.

3.2.5.2.1 Graph Definition

The graph field is defined as a field on a form in the same way as other fields. The graph field must also appear in a CREATE GRAPH statement. The size and location are required parts of the graph field definition. All other attributes of the graph field definition are optional.

3.2.5.2.2 Additive Versus Absolute Display

When data are displayed using bar charts or x-y plots, more than one dependent data set may be plotted in a given graph. Two or more curves are distinguishable by color or linestyle for example. The ordinate values can be measured on the vertical axis either relative to the horizontal axis or relative to a curve already displayed. The former method is absolute display

and the latter additive display. If the graph is a bar chart, the bars will be displayed side by side if the display is absolute and displayed stacked vertically if the display is additive. By default displays will be absolute.

3.2.5.2.3 Axis Information

Axis labels and tick mark labels can be specified independently for the horizontal and vertical axes. The text in these labels can be defined to have any available font, color, and size (in units of the terminal's standard character height and width). By default text will have a size of 1 by 1 and a simple stick figure font; the color will be the default contrasting color to the background, e.g. white on black. The color of the axis itself including tick marks can also be specified and may differ from the color of the labels.

The tick mark clause specifies the number of major tick marks and optionally the number of minor divisions between each pair of major tick marks. The tick mark label string comprises a sequence of substrings each delimited by a chosen character not appearing elsewhere in the string. If the number of substrings is fewer than the number of major tick marks, then the tick mark label string is reused from the beginning.

If no label string is defined for an axis, the associated item field name is used as the label. In the case that more than one item field is associated with a vertical axis, the name of the first such item field is used. If no tick mark label string is defined, then the axis is divided into a number of intervals which yields nice tick mark label values. Nice values are defined to be multiples by powers of ten of values in the set { .1, .2, .25, and .5}.

The lower and upper limits and the scale (linear or logarithmic) of the axes can be specified also. If the range of the data exceeds the limits specified, then data outside the range are not displayed. Unspecified limits are set by default to give the smallest range which will span the range of the data such that the tick mark labels can either be those given by the user or can be ones in the set of nice values as defined in the paragraph above. If the scale is not defined to be logarithmic, then by default it is linear.

3.2.5.2.4 Background Color

The background of the graph field can be defined to have an

available color.

3.2.5.2.5 Curve Information

For x-y plots and in most instances for bar charts several attributes of the curves or bars may be specified including the color of the curve or bars as well as the color or shading pattern of the area below the curve or bars. On devices supporting the feature the linetype (solid, dashed, etc.) and linewidth may also be specified. For x-y plots symbols may optionally be placed at the data point locations either in addition to or instead of the curve. By default data points are connected by solid line segments in the order of occurrence of the source item field values and no data point symbols appear. The symbols may be chosen from a catalog of predefined symbols and will include minimally the dot, plus sign, asterisk, cross or x, and circle. By default no shading occurs below curves or bars and the color is the background color.

3.2.5.2.6 Grids

Bar charts and x-y plots can be defined to have grids superimposed upon them. By default no grid is displayed. A grid will connect major tick marks in both directions. If minor tick marks have been specified, then a fine grid will connect minor tick marks. If minor tick marks have been defined for only one axis, then fine grid lines will appear perpendicular to only that axis.

3.2.5.2.7 Legends

A graph can be defined to have a legend in a specified location. By default no legend will appear. The legend can also optionally be circumscribed by a box. The location value is the location of the upper left hand corner of the legend relative to the graph field. The legend entry is defined in the AXIS or PIE clauses of the CREATE GRAPH statement which may also contain specifications of font, size, and color attributes for the legend entry text.

3.2.5.2.8 Margins

The amount of space surrounding the graph which is delimited by the axes or by the pie is adjustable. This space can be set by the MARGIN keyword in the GRAPH field definition. The number given is the percentage of the corresponding dimension of the graph field. By default all margins are 10

percent.

3.2.5.2.9 Pie Charts

There are several language features which apply only to pie charts. Some pie chart attributes are associated with individual segments. The segments are numbered sequentially from 1 beginning at a horizontal radius vector to the right.

The explosion factor which is applied on a segment by segment basis is the percentage of the radius of the pie by which a segment is projected radially outward. By default no segments are exploded.

The data source for a pie chart is a single item field. The segments correspond in order with the elements of the item field. The item field values and/or their corresponding percentages of the whole may be displayed for each segment either inside or outside the segment. The location is the same for all segments, but quantities and percentages need not both be inside or outside. By default neither the quantities nor the percentages will appear.

As for other text, color, size, and font may be defined for quantities, percentages, segment labels, and legend entries.

Finally, each segment may be individually colored or shaded with a pattern. By default no pattern will appear within the segments and the interior color will be the background color.

3.2.5.2.10 Text

Text may be included anywhere within the graph field for titles, annotation, etc. via the PROMPT clause which is similar to the PROMPT clause generally available within FDL except that here the text attributes include font and size and, if supported by the device, color.

3.2.5.3 Stretchy Lines

The developer using this CPCI shall be able to define extensible lines associated with forms and items. The starting and ending locations mean the centers of the given character positions. By using relative positions for the ending locations of lines, the developer can define lines which effectively stretch to correspond to the length or width of the associated field as it expands at run time.

The exact representation of the lines is device dependent. On a particular device the mechanism with the highest resolution will be the one implemented. If the device only supports typewriter graphics, then the dash will be used to construct horizontal lines and the vertical bar (if available) or the exclamation point will be used to construct vertical lines. At the juncture or intersection of two lines, a plus sign will replace the vertical and horizontal characters.

3.2.6 Security and Access Control

Security and access control is provided through the mechanism of conditional forms and triggers sensitive to particular item field values. Thus, the functionality offered to a particular user can be restricted by not displaying a form containing options for certain database operations or by not enabling certain function keys. Further, other forms informing the user of restrictions on his access can be displayed instead.

3.2.7 Conditional Actions

Conditional actions are those triggered by the occurrence of an event defined in an ON statement. Any number of actions can be triggered by a single event. They involve both form processor actions and database transactions. In fact, it is primarily the conditional actions which determine the course of the execution of the application.

3.2.7.1 Form Processor Actions

3.2.7.1.1 PRESENT Form

This action causes the instantiation of the specified form and subsequently its output to the user's terminal. This action has two varieties depending on whether or not the form is a subform of some other form defined in the application definition. First level forms, i.e. forms that are not subforms are simply PRESENTed. If a subform is to be PRESENTed, it must be declared to be CONDITIONAL and it must be PRESENTed in a window defined as part of the containing form. In this way a subform can conditionally appear depending upon some triggered event, or one of several alternative subforms can be displayed perhaps as an item field has one of several alternative values.

3.2.7.1.2 SET Option

This option allows the setting of an item field to a specified integer or string constant value.

3.2.7.1.3 HELP String or Form

This action allows the developer to output help information to the user either unsolicited or in response to some user triggered event. The help can be simply a message line prompt or a whole form.

3.2.7.1.4 EXIT

This action allows the user to terminate the application.

3.2.7.2 Database Transactions

It is by means of the conditional action lists attached to ON conditions that database operations occur.

3.2.7.2.1 Delete

This action will cause commencement of a deletion operation against the database which will be controlled as discussed above in section 3.2.3.1. The control is specified in the delete action statement and may refer to one or more form item field values.

3.2.7.2.2 Insert

This action will cause one or more records to be appended to the end of the specified table.

3.2.7.2.3 Modify

This action will begin a transaction to modify one or more records in the database identified by the attached WHERE clause as described in section 3.2.3.3.

3.2.7.2.4 Select

This action retrieves one or more records from the CDM according the criterion given in the attached WHERE clause.

3.2.8 Data Structures

Depicted in Figure 3.2 below are the data structures particular to the Rapid Application Generator. Many data structures already included in the Forms Processor are omitted for brevity. The data structures will be created by FLAN and will be used by the RAP to create the source code for the application.

3.2.9 Generated Application Structure

The start up actions are performed when the procedure is started. The actions associated with a condition are added to a priority queue by the procedure add_actions. Conditions form a nested if-then-else group. Each PICK and set item condition is checked by a procedure trign, where n is the number of the condition. All change conditions are checked by the procedure chkcng. The main program outline follows:

```
add_actions(startup);    /* add startup trigger's actions to
          list */
       while (actions)
          perform action;

       while (true)
          {
          /* form_name is the form in 'screen;' */

          if (strcmp(form_name, "form1") == 0) /* restore
                              form's contents */
          pdata(path1, form1c, rcode);
          if (strcmp(form_name, "form2") == 0)
          pdata(path2, form2c, rcode);
          ...

          oiscr("screen", &pfkey, rcode);
```

```
/* assign current to previous and get new current */

        if (strcmp(form_name, "form1") == 0)
           {
           memcpy(form1p, form1c, sizeof form2);
           gdata(&current, path1, form1c, rcode);
           }
        if (strcmp(form_name, "form2") == 0)


           {
           memcpy(form2p, form2c, sizeof form2);
           gdata(&current, path1, form2c, rcode);
           }
        ...

/* check a trigger for true */
        if (trig0()) add_actions(0);
        else if (trig1()) add_actions(1);
        . . .
        else if (trign()) add_actions(n);
        else chkcng()                  /* check for changes */

        while (actions)
           perform action;
        }
```

Figure 3-2 Rapid Application Generator Data Structures

3.3 Performance Requirements

3.3.1 Programming Methods

A C program will be used to input the data structures created by compiling the .FDL source file with FLAN and the external schema data structures from the CDM and output the C and COBOL code which define the application.

3.3.2 Program Organization

The Rapid Application Generator will be developed in a modular way that will allow as great a use as possible of existing routines for progressive extension of functionality.

3.4 Data Base Requirements

The Rapid Application Generator accesses the CDM data dictionary information (the data about the data stored in the CDM) to generate necessary application code for the External Schemas. It also accesses the Application Definition Form Processor structures to generate necessary application code for processing the Presentation Schemas.

3.4.1 Sources and Types of Input

CDM - Provides characteristic information about the External Schema items being used by the application. The machine representation, the size, and number of positions after the decimal point are retrieved from the CDM by the Rapid Application Generator.

Application Definition Objects - Provide characteristic information about the Presentation Schema items. The machine representation and the picture format are retrieved from the Application Definition Objects by the Rapid Application Generator.

3.4.2 Destination and Types of Output

Generated External Schema Cobol record structures - The Rapid Application Generator uses the information from the CDM to create the Cobol record structures for the External Schemas being used by the application program. These structures are part of the generated application code source file.

Generated Presentation Schema Cobol record structures - The Rapid Application Generator uses the information from the Application Definition Objects to create the Cobol record structures for the Presentation Schemas being used by the application program. These structures are part of the generated application code source file.

Generated Machine Representation Conversion code - The Rapid Application Generator uses information in the CDM and Application Definition Objects to create the correct code to go from the External Schema to the Presentation Schemas and vice versa. This code is part of the generated application code source file.

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definition

"Testing" is a systematic process that may be preplanned and explicitly stated. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error.

"Antibugging" is defined as the philosophy of writing programs in such a way as to make bugs less likely to occur and when they do occur, to make them more noticeable to the programmer and the user. In other words, as much error checking as is practical and possible in each routine should be performed.

4.2 Computer Programing Test and Evaluation

The quality assurance provisions for test consist of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests are performed by the design team. Structured design, design walk-through and the incorporation of "antibugging" facilitate this testing by exposing and addressing problem areas before they become coded "bugs".

The integration testing entails generating from an ADL definition an application comprising C and COBOL code, compiling the C portion, precompiling the COBOL portion with CDMP, compiling the resultant COBOL program, installing the application on the IISS UIS, and producing an interactive database application using a CDM data base via the NTM.

Each function is tested separately, then the entire sub-system is tested as a unit. Development and testing are done on the Air Force VAX.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the ICAM Integrated Support System (IISS) Test Bed site located at General Electric, Albany, New York. The software associated with each CPCI release is delivered on a media which is compatible with the IISS Test Bed. The release is clearly identified and includes instructions on procedures to be followed for installation of the release. Integration with the other IISS CPCI's is done on the IISS Test Bed on a scheduled basis.

APPENDIX A

APPLICATION GENERATOR LANGUAGE SYNTAX

The complete ADL Syntax is listed in this section.

Application Definition

```
CREATE APPLICATION application_name

     [ Form_Definition ] ...

     [ Condition_Definition ] ...

     [ Graph_Definition ] ...
```

Form Definition

```
CREATE FORM form_name

     [ CONDITIONAL ]

     [ BACKGROUND { WHITE  } ]
                  { BLACK  }
     [ KEYPAD ( KEY n = function_name ... ) ]

     [ PROMPT Location prompt_string ] ...

     [ Field_Definition ] ...
```

Field Definition - Items

```
ITEM item_name [ Repeat_Spec ]

     [ Location ]

     [ SIZE cols [ BY rows ] ]

     [ VALUE {expression    } ]
             {INDEX(field_name)}
             {'._DATE'         }
             {'._TIME'         }

     [ NODUP ]

     DISPLAY AS { INPUT }
                { OUTPUT }
                { HIDDEN }
                { TEXT   }

     [ DOMAIN ([LEFT ] [UPPER] [ PICTURE picture_spec ])]
              [RIGHT] [LOWER]

              [ MUST ENTER ] [ MUST FILL ] [ NUMERIC ]
              [ MAXIMUM int ] [ MINIMUM int ]


     [ PROMPT Location prompt_string ...]
```

```
[ HELP { string     } ]
       {form_name   }
       {APPLICATION}

[ LINE FROM Location-1 TO Location-2 ... ]
```

Field Definition - Forms

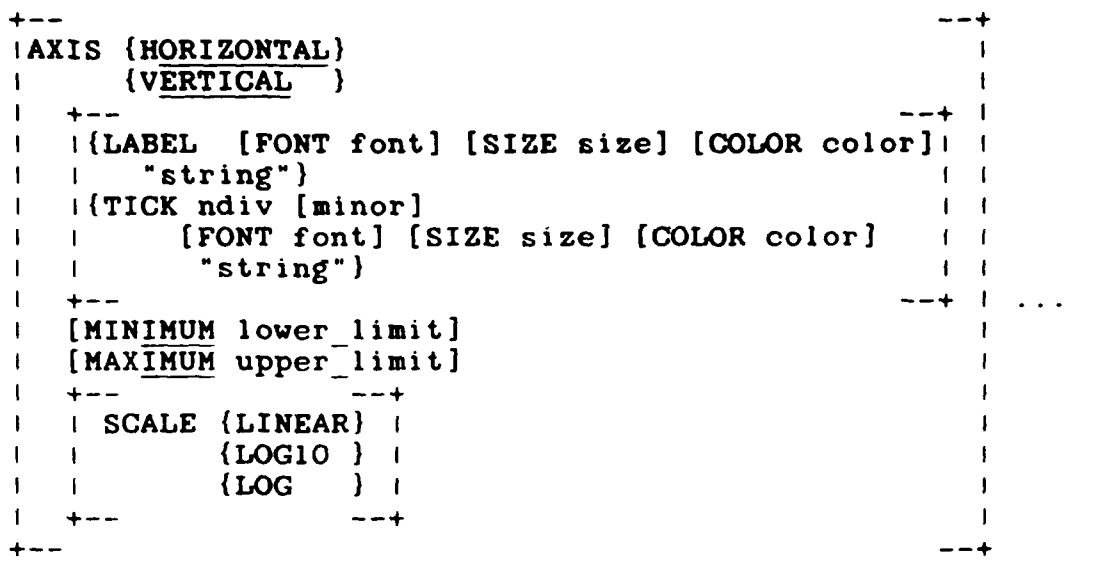
```
FORM form_name [ Repeat Spec ]

     Location

     SIZE cols [ BY rows ]

     [ PROMPT Location prompt_string ] ...

     [ LINE FROM Location-1 TO Location-2 ... ]
```

Field Definition - Windows

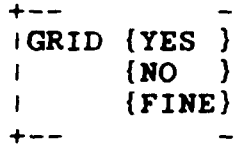
```
WINDOW window_name [ Repeat Spec ]

     Location

     SIZE int [BY int ]

     BACKGROUND { BLACK   }
                { WHITE   }

     [ PROMPT Location prompt_string ] ...
```

Field Definition - Graphs

```
     GRAPH graph_name

     +--          --+
     |{ABSOLUTE}   |
     |{ADDITIVE}   |
     +--          --+
```

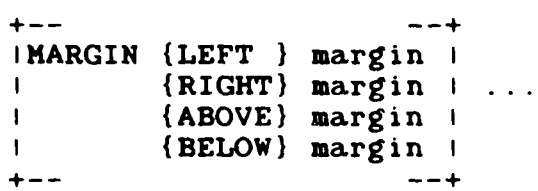


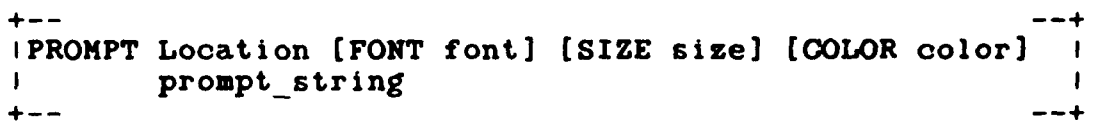
[BACKGROUND color]



[LEGEND Location [BOX]]

Location





SIZE cols BY rows

```
+--          --+
|TYPE {BAR }  |
|     {PIE }  |
|     {PLOT}  |
+--          --+
```

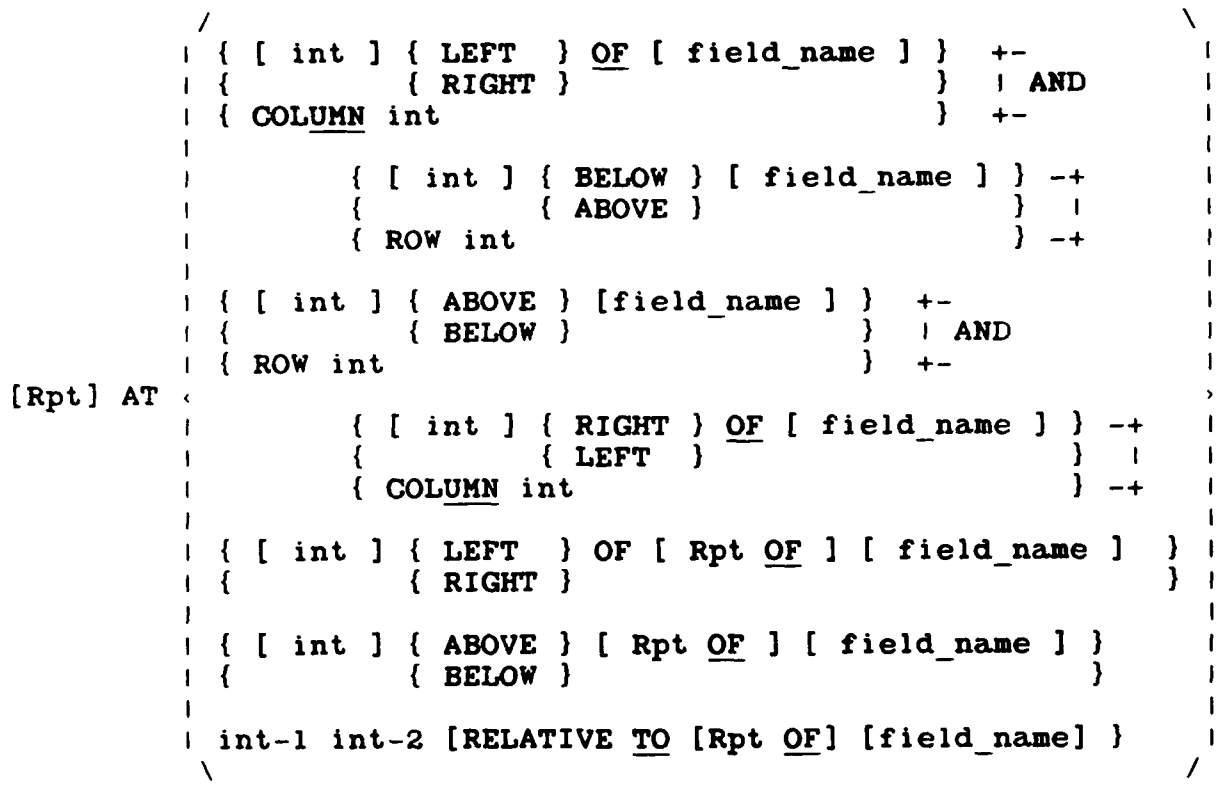
Field Definition - Icons

```
ICON icon_name
     +--                              --+
     | { USING item_field_name }       |
     | { BIT (bit_string)      }       |
     | { CHAR (char_string)    }       |
     +--                              --+
     Location
     SIZE size
     [DISPLAY AS {OUTPUT}]
                 {TEXT  }
     [PROMPT Location prompt_string]
```

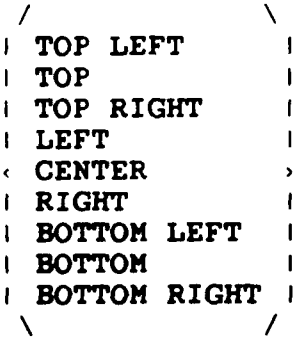
Repeat Spec

```
+--                                                                  --+
| ({ * } { HORIZONTAL } [ WITH int SPACES ] [ , ... ] )              |
|  {int} { VERTICAL   }                                              |
+--                                                                  --+
```

Location



Rpt



Condition Definition

```
      /                                                  \
     | (CHOOSE icon_name) Condition_Action ...            |
     | (OVERFLOW BY field_name) Condition_Action ...      |
     | (CHANGE item_name) Condition_Action ...            |
     | (item_field {= } value) Condition_Action ...       |
 ON  <              {!=}                                  >
     | [(PICK function_name) Condition_Action ...         |
     | [(PICK function_name & CURSOR IN field)            |
     |        Condition_Action ...                        |
     | (STARTUP) Condition_Action ...                     |
      \                                                  /
```

Condition Action

```
     /                                  \
    | PRESENT form_name [IN window ]     |
    | SET item_name = {string }          |
    |                 {integer}          |
    | HELP { string    }                 |
    <      { form_name }                 >
    | Delete_action                      |
    | Insert_action                      |
    | Modify_action                      |
    | Select_action                      |
    | EXIT                               |
     \                                  /
```

Delete Action

DELETE FROM table WHERE Predicate

Insert Action

INSERT INTO table (col_spec [...]) VALUES value_list

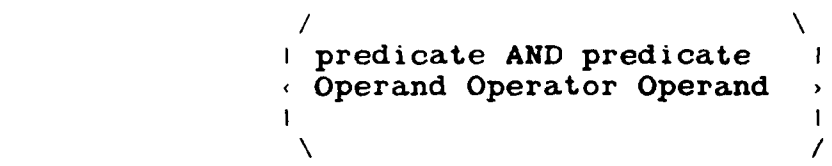
Modify Action

MODIFY table [USING table] SET column_assignment_list
WHERE Predicate

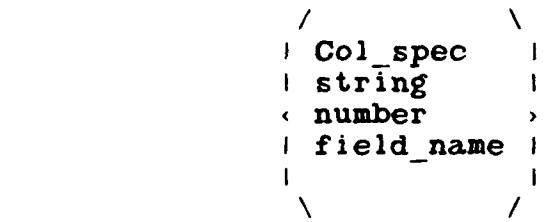
Select Action

```
SELECT qualified_name = col_spec ...
       [ DISTINCT ]
       [ FROM table [ abbreviation ] ,... ]
       [ WHERE Predicate ]
       [ ORDER BY Col_spec { ASCENDING } ... ]
                           { DESCENDING}
       [ '{' select_action ... '}' ]
```

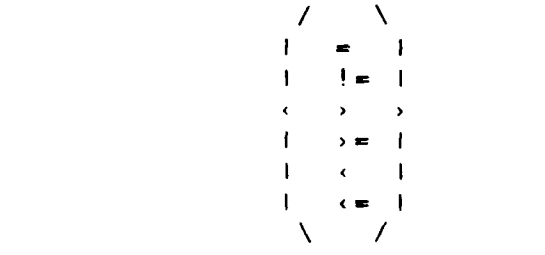
Predicate



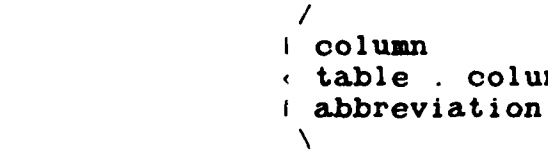
Operand



Operator

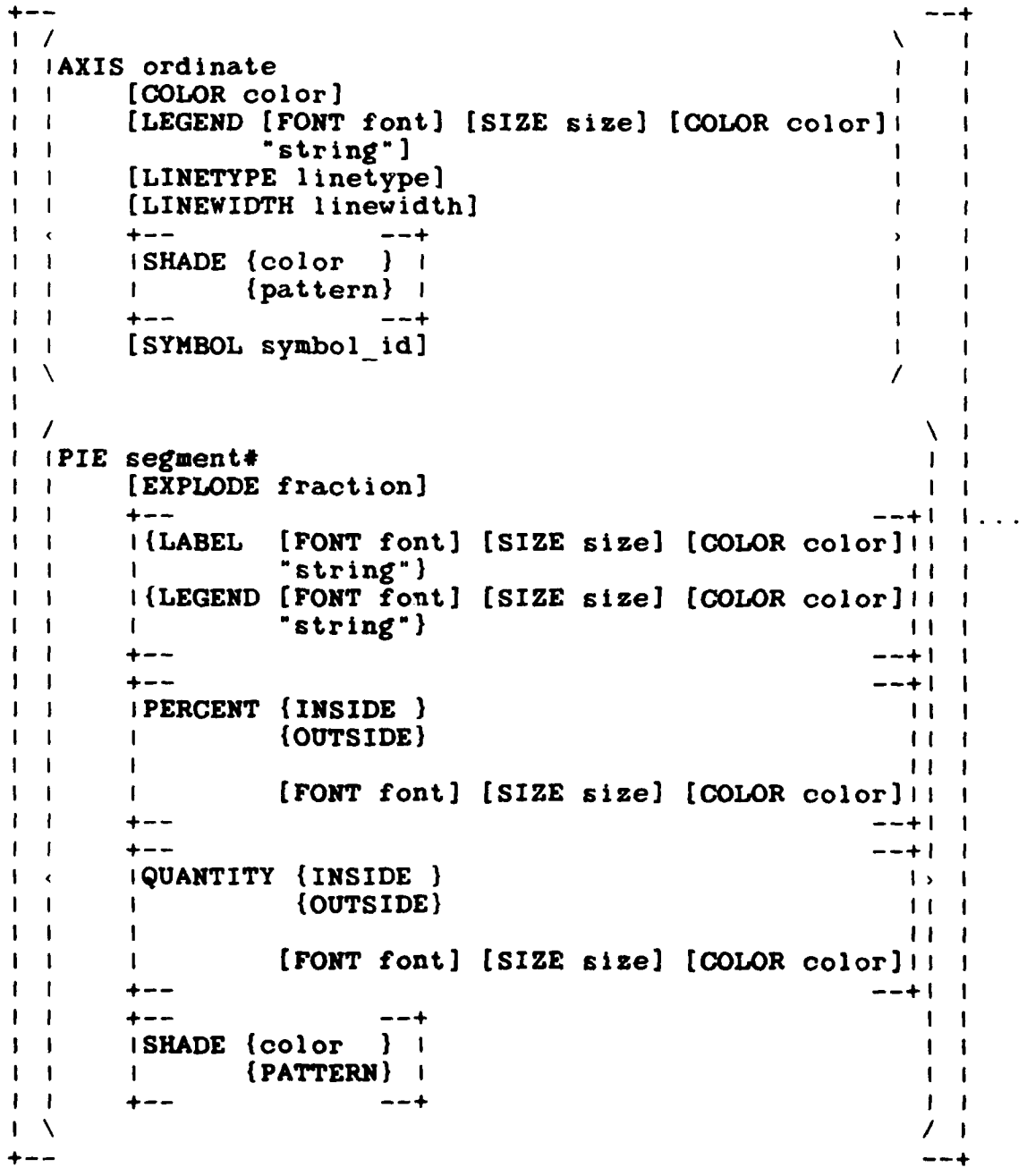


Col_spec



Graph Definition

```
CREATE GRAPH graph_name
     USING ordinate [[VERSUS] abscissa] ...
```



U.S.Government Printing Office 1987 748 061 60058

END

8-87

DTIC